bcopy()

Be careful with buffer size and termination

Sean Barnum, Cigital, Inc. [vita[1]]

Copyright © 2005 Cigital, Inc.

2005-10-03

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 5246 bytes

<table>
<tr><td><b>Attack Categories</b></td><td>• Malicious Input
• Denial of Service</td></tr>
<tr><td><b>Vulnerability Categories</b></td><td>• Buffer Overflow
• No Null Termination</td></tr>
<tr><td><b>Software Context</b></td><td>• String Management</td></tr>
<tr><td><b>Location</b></td><td>• strings.h</td></tr>
<tr><td><b>Description</b></td><td>bcopy (const void *src, void *dest, int n) - copies the first n bytes of the source string src to the destination string dest.
There are many generic types of errors that can apply to bcopy(). These include
• mis-specifying the size of a buffer or the amount of data to be written. Off-by-one errors are common.
• failing to plan for correct behavior when input is larger than expected.
• assuming the wrong semantics for a parameter that controls data transfer and prevents buffer overflows. Because various functions use the buffer size, buffer size minus one, the remaining space in the buffer, etc., it is important to understand the bounding semantics for each function.
In the context of strings:
• Failing to allow space for a terminating null character.
• Failing to ensure that a terminating null character is present; many standard functions consistently experience this failure.</td></tr>
<tr><td><b>APIs</b></td><td><table><tr><th>FunctionName</th><th>Comments</th></tr><tr><td>bcopy()</td><td>This function is deprecated -- use memcpy in new programs.</td></tr></table></td></tr>
</table>

[1] http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

<table>
<tr><td><b>Method of Attack</b></td><td>Bounds checking and off-by-one errors create opportunities for buffer overflow or denial of service attacks.</td></tr>
<tr><td><b>Exception Criteria</b></td><td></td></tr>
<tr><td><b>Solutions</b></td><td>
<table>
<tr><th>Solution Applicability</th><th>Solution Description</th><th>Solution Efficacy</th></tr>
<tr><td>Always</td><td>Always use #define or other const for declaration of size</td><td>Effective to the degree that consistent care is used.</td></tr>
<tr><td>Always</td><td>Always use SAME #define or const when checking bound sizes</td><td>Effective to the degree that consistent care is used.</td></tr>
<tr><td>Always</td><td>For strings, use (buffer size) - 1 to ensure space to put terminating \0</td><td>Effective to the degree that consistent care is used.</td></tr>
<tr><td>Always</td><td>For strings, always write a \0 to upper bound the buffer after processing a string</td><td>Effective to the degree that consistent care is used.</td></tr>
<tr><td>Always</td><td>Do a bounds check to verify that the buffer you are passing in is as big as you say and that it is big enough to hold the new contents. Verify that the returned buffer is null terminated.</td><td>Effective to the degree that consistent care is used.</td></tr>
<tr><td>Always</td><td>Identifying or writing safer versions of utility functions which incorporate checks, and then using these safer functions consistently</td><td>Effective to the degree that consistent care is used.</td></tr>
</table>
</td></tr>
</table>

	improves safety.
Signature Details	

Examples of Incorrect Code	`int count[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};`
	`int ids[3]={0, 1, 2};`
	`bcopy(count, ids, 5);`

Examples of Corrected Code	`const int SOURCE_BUFFER_SIZE = 10;`
	`const int DESTINATION_BUFFER_SIZE = 3;`
	`int count[SOURCE_BUFFER_SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};`
	`int ids[DESTINATION_BUFFER_SIZE] = {0, 1, 2};`
	`bcopy(count, ids, DESTINATION_BUFFER_SIZE*sizeof(int)); /* limit number of integers to be copied */`

Source References	• http://security-protocols.com/unixmanpages/bcopy.3.html • ITS4 Source Code Vulnerability Scanning Tool[3]
Recommended Resources	
Discriminant Set	**Operating System**: • UNIX (All) **Languages**: • C • C++

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@cigital.com[1].

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

[1] mailto:copyright@cigital.com